

CLAIMS

What is claimed is:

- 5           1. A method of computation comprising:
  - a) performing a critical task using an operation that is speculative while a condition of a processor used for performing said critical task is unknown;
  - b) in parallel with a), determining said condition;
  - 10          c) if said condition is as expected, committing a first result from said performing said critical task; and
  - d) if said condition is not as expected, allowing said first result to benignly fail, changing said condition to be as expected, re-performing said critical task using said operation, and committing a second result  
15          from said re-performing said critical task.
2. The method of computation as described in Claim 1, wherein said operation is a load.
- 20          3. The method of computation as described in Claim 1, wherein a) further comprises:
  - assigning a speculative attribute to said operation.
4. The method of computation as described in Claim 1, wherein  
25          said condition is an enablement state of virtual memory.
5. The method of computation as described in Claim 4, wherein b) comprises:
  - determining, in parallel to a), if data translation is enabled,
  - 30          wherein when data translation is enabled said condition is as expected, and when data translation is disabled said condition is not as expected.
6. The method of computation as described in Claim 1, wherein d) comprises:

setting preconditions in virtual memory to change said condition to be as expected.

7. The method of computation as described in Claim 1, wherein a)  
5 further comprises:  
receiving an interruption at an interruption handler to perform said critical task.

8. The method of computation as described in Claim 7, wherein  
10 said interruption handler is a first-level interruption handler operating in a lightweight interruption environment.

9. The method of computation as described in Claim 1, wherein c)  
further comprises:  
15 storing said first result in a virtual memory address; and  
wherein d) further comprises:  
storing said second result in said virtual memory address.

10. The method of computation as described in Claim 1, wherein  
20 d) further comprises:  
assigning a non-speculative attribute to said operation,  
duplicating code used to execute a) forming a duplicated code, and re-performing said critical task using said duplicated code.

25 11. A method of computation comprising:  
a) receiving an interruption from an application to perform a critical task; and  
b) incorporating speculative features of a processor to perform said critical task, that references virtual memory addresses that are  
30 known and valid, but a condition of a processor used for performing said critical task is unknown; and  
c) using a speculative load to perform said critical task.

12. The method of computation as described in Claim 11, wherein  
35 c) further comprises:

c1) performing said critical task;  
 c2) in parallel with c1) determining said condition;  
 c3) if said condition is as expected, committing a first result from  
 said performing said critical task; and

5 c4) if said condition is not as expected, changing said condition to  
 be as expected, and re-performing said critical task successfully while  
 allowing said first result to benignly fail.

13. The method of computation as described in Claim 12, wherein  
 10 c4) further comprises:  
 committing a second result from said re-performing said critical  
 task.

14. The method of computation as described in Claim 11, wherein  
 15 said interruption is handled by an interrupt handler.

15. The method of computation as described in Claim 12, wherein  
 c3) further comprises:

determining virtual memory is enabled such that said condition is  
 20 expected; and

wherein c4) further comprises:

determining virtual memory is disabled such that said condition  
 is not as expected.

25 16. A method of computation comprising:

a) receiving an interruption from an application to perform a  
 critical task;

b) performing said critical task using a load that is speculative  
 while a condition of virtual memory is unknown;

30 c) in parallel with b), determining said condition;

d) if virtual memory is enabled, committing a first result from  
 said performing said critical task;

e) if virtual memory is disabled, enabling said virtual memory in  
 order to re-perform said critical task successfully while allowing said  
 35 first result to benignly fail since said load is speculative.

17. The method of computation as described in Claim 16, wherein  
e) further comprises:

e2) re-performing said critical task using said load; and

5 e3) committing a second result from said re-performing said  
critical task.

18. The method of computation as described in Claim 16, wherein  
a) further comprises:

10 assigning a speculative attribute to said load.

19. The method of computation as described in Claim 16, wherein  
an interruption handler receives said interruption and performs said  
critical task.

15 20. The method of computation as described in Claim 16, wherein  
said interruption handler operates in a lightweight interruption  
environment.

20 21. The method of computation as described in Claim 16, wherein  
d) further comprises:

d1) assigning a non-speculative attribute to said load, forming a  
non-speculative load;

d2) duplicating code used to execute b), forming a duplicated code;

25 and

d3) re-performing said critical task with said duplicated code  
using said non-speculative load.

22. A computer system comprising:

30 a processor; and

a computer readable memory coupled to said processor and  
containing program instructions that, when executed, implement a  
method of computation comprising:

a) performing a critical task using an operation that is speculative while a condition of a processor used for performing said critical task is unknown;

b) in parallel with a), determining said condition;

5 c) if said condition is as expected, committing a first result from said performing said critical task; and

d) if said condition is not as expected, allowing said first result to benignly fail, changing said condition to be as expected, re-performing said critical task using said operation, and committing a second result  
10 from said re-performing said critical task.

23. The computer system as described in Claim 22, wherein said operation is a load.

15 24. The computer system as described in Claim 22, wherein a) in said method further comprises:

assigning a speculative attribute to said operation.

20 25. The computer system as described in Claim 22, wherein said condition is an enablement state of virtual memory.

26. The computer system as described in Claim 25, wherein b) in said method comprises:

determining, in parallel to a), if data translation is enabled,  
25 wherein when data translation is enabled said condition is as expected, and when data translation is disabled said condition is not as expected.

27. The computer system as described in Claim 22, wherein d) in said method comprises:

30 setting preconditions in virtual memory to change said condition to be as expected.

28. The computer system as described in Claim 22, wherein a) in said method further comprises:

receiving an interruption at an interruption handler to perform said critical task.

29. The computer system as described in Claim 28, wherein said  
5 interruption handler is a first-level interruption handler operating in a lightweight interruption environment.

30. The computer system as described in Claim 22, wherein c) in said method further comprises:

10 storing said first result in a virtual memory address; and  
wherein d) further comprises:  
storing said second result in said virtual memory address.

31. The computer system as described in Claim 22, wherein d) in  
15 said method further comprises:

assigning a non-speculative attribute to said operation,  
duplicating code used to execute a) forming a duplicated code, and re-  
performing said critical task using said duplicated code.

20